IoT Database Forensics: An Investigation on HarperDB Security

Robert Marsh
Sana Belguith
Tooska Dargahi
R.Marsh3@edu.salford.ac.uk
S.Belguith@salford.ac.uk
T.Dargahi@salford.ac.uk
School of Computing, Science and Engineering, University of Salford, Manchester, UK
R.Marsh3@edu.salford.ac.uk,S.Belguith@salford.ac.uk,T.Dargahi@salford.ac.uk

ABSTRACT

The data that are generated by several devices in the IoT realm require careful and real time processing. Recently, researchers have concentrated on the usage of cloud databases for storing such data to improve efficiency. HarperDB aims at producing a DBMS that is relational and non-relational simultaneously, to help journeymen developers creating products and servers in the IoT space. Much of what the HarperDB team has talked about has been achieved, but from a security perspective, a lot of improvements need to be made. The team has clearly focused on the problems that exist from a database and data point of view, creating a structure that is unique, fast, easy to use and has great potential to grow with a startup. The functionality and ease of use of this DBMS is not in question, however as the trade-off triangle to the right suggests, this does entail an impact to security. In this paper, using multiple forensic methodologies, we performed an in-depth forensic analysis on HarperDB and found several areas of extreme concern, such as lack of logging functionalities, basic level of authorisation, exposure of users' access rights to any party using the database, There had to be a focus on preventative advice instead of reactive workarounds due to the nature of the flaws found in HarperDB. As such, we provide a number of recommendations for the users and developers.

CCS CONCEPTS

• **Security and privacy → Distributed systems security**; **Database activity monitoring**.

KEYWORDS

HarperDB, IoT, Security, IoT Databases

ACM Reference Format:
Robert Marsh, Sana Belguith, and Tooska Dargahi. 2019. IoT Database Forensics: An Investigation on HarperDB Security. In *Proceedings of ACM International Conference on Future Networks and Distributed Systems (ICFNDS).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

ICFNDS, July 2019, Paris, France

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Organisations are known for their zealous nature towards storing data in relational databases. However, thanks to advances in storage technology (being larger and cheaper) more data of customers is being stored in the cloud, anything from photos, to text, to audio and even videos [12, 16]. The problem is, the data collected is no longer relational - Amazon and Google have adopted a new structure called NoSQL to meet this new requirement [15, 19]. This type of database has been around since the 1960s, however, it is only in the last decade that we have seen market traction and the trend towards database, such as HarperDB [5], that are both capable of storing relational and non-relational data together [11, 19].

The huge amount of data that is generated by Internet of Things (IoT) devices should be stored and processed securely and efficiently [13, 14]. According to Forbes, HarperDB is used by IoT devices, "machines out in the extremities of a network, on an aircraft, on an oil rig, in any remote location even in cities" [1]. Thanks to the design and size of HarperDB, the same database system will exist on an IoT device and in the cloud - the main difference being what is contained within the databases. Only a small section of data may be relevant to the IoT device, whereas the cloud database will store all data from all devices. The cloud database will likely be running big data analytics on a vast dataset from thousands of IoT devices relaying back the most important information for each specific 'thing' to store [4]. This throws up serious questions in relation to a forensic investigation, which we are going to address in this paper. If the database is the same for these two very different systems and scenarios, how security and auditing will work adequately in HarperDB? Considering that IoT device tampering out in the extremities of a network has a very different set of security implications to that of a cloud server running in a controlled environment.

HarperDB has been created to address the complexity and expense of current database systems. Aimed at software developers of all skill levels, the database can be built and scaled without any prior knowledge or understanding and without sacrificing performance [3]. It has been marketed as a "true native high transactional NoSQL and SQL" single database solution [4]. From a layman's perspective, HarperDB would be a very enticing platform. Marketing aimed at this segment should invariably mean that the database system offers great out-of-the-box security features and measures. This market segment is likely not very knowledgeable about the implications and risks of having an insecure database and how to

enhance the security of these databases to be resistant against potential adversaries. It would be expected that HarperDB offers some tools, visual aids and guides that help their users with the security aspects relevant to the database system and its implementation.

In this paper, we perform an in-depth forensic analysis of HarperDB 1.2.004, while considering a combination of two methodologies: Database Forensics (DBF) which introduces relevant steps to carry out databases forensic investigation and Common Database Forensic Investigation Process (CDBFIP) detailing suitable steps to investigate IoT environments. We analyse and test attribution, traces and security features in HarperDB. The results of our thorough investigation show that there are some security flaws which require to be addressed.

Paper Organisation –The reminder of this paper is organised as follows. Section 2 reviews related work. Section 3 explains the methodology we followed to carry out HarperDB investigation. Section 4 presents HarperDB structure and its main functionalities while Section 5 analyses HarperDB security features and explain potential existing vulnerabilities. A set of best practices and advices for developers and end-users dealing with HarperDB as well as some mitigation techniques are presented in Section 6 before concluding the paper in Section 7.

2 RELATED WORK

Several research papers have studied and assessed security features in most known Database Management Systems (DBMS). Hauger [18] concluded in their paper that NoSQL databases are known for the storage of sensitive data and this has made them a target for all types of hackers. However, there is a lack of "default security measure"âĂŹ; meaning that specific security features are not always enabled automatically in regards to logging, authentication, authorization and access control. It appears to be common practice for NoSQL databases to have these security features disabled by default. Ultimately, it was concluded that many of these new database systems do not grasp the basics of what is needed for forensic attribution.

One of the most important areas of a digital forensics investigation is the ability to reconstruct database data from historical logs. Many database management systems offer logging capabilities but do not explain properly what they are. It is important that an ideal logging level exists and information on how to set this up is included with the DBMS [8]. Adedayo [8] found that the default settings in many database management systems are set incorrectly for a true forensic analysis with the ability for reconstruction to take place. The paper noted that several main elements that require logging such as data modification queries, metadata changes, access information, timestamps and users, changes in privileges, system failures and errors and archived logs. In [20], Shahriar studied security vulnerabilities of both SQL and NoSQL databases. They mainly investigated Cassadra [2] and MongoDB [6] databases which are considered the most famous NoSQL databases. Indeed, they identified several security issues in both studied databases. For instance, MongoDB as well as Cassadra do not support encrypted client-server communication neither files encryption and authentication and authorisation are disabled by default. Gupta [17] analysed security features in NoSQL databases. They have been mainly interested in investigating MongoDB. The authors identified several vulnerabilities and potential attacks such as authentication bypass, Blind NoSQL Injection, denial of service,

The aforementioned research works have been mainly dealing with SQL and NoSQL databases. However, to the best of our knowledge, there is no security assessment research on HarperDB to identify security vulnerabilities, yet. In this paper, we establish an investigation on HarperDB to point out the existent security vulnerabilities and provide some recommendations.

3 INVESTIGATION METHODOLOGY

Database Forensic (DBF) is a process to identify, collect, preserve, analyse, reconstruct and document digital evidence created by an attack on a Database Management System (DBMS) [10]. As this paper deals with a platform that has also an IoT aspect, this environment goes beyond that of most DBMS, which implies that DBF process cannot cover all the features of HarperDB. Therefore, we will apply the Common Database Forensic Investigation Process (CDBFIP) [9] introducing a set of best practices and processes that are most suitable for IoT forensics investigation.

Using a mixture of both CDBFIP [9] and DBF Metamodel (DBFM) [10], we investigate HarperDB through the following three main steps: identification, collection and analysis. In the identification stage, the environment set up including the required software and data creation is performed while describing the structure and concept of HarperDB, as well as its settings and functionalities.

The collection phase deals with identifying the structure of stored data and their accessibility. Furthermore, this phase determines logs settings, as well as their archiving techniques. Security features applied on data such as encryption, authorisation and authentication are also studied.

Finally, the analysis phase reports data trails structures, logged breadcrumb and archive security. It analyses main security features and their influence on the security of data stored in HarperDB. This phase also identifies anomalies found during the identification and collection stages.

Each of the investigation steps presented above link back to either the CDBFIP [9] or the DBFM [10] frameworks. We slightly adapted the models produced in these papers for the experimental process implemented in our paper. Unlike other papers that focus on helping analysts detect and reverse tampering, this paper is aiming to be a more proactive piece of literature rather than reactive. If the DBMS being worked with is flawed or set up incorrectly, it does not matter how good a forensic analyst is or how well they implement a forensic framework, their job will be impossible. The aim of this paper is to work towards creating security advice for both the teams behind this DBMS product and the developers who use it.

4 HARPERDB STRUCTURE AND FUNCTIONALITIES

In this section, we detail the main structure of HarperDB and we report its main functionalities.

4.1 HarperDB Structure

The overview of the HarperDB layout can be seen in Figure 1. HarperDB relies on SmartGit[1] that is a Git client for Windows, Mac and Linux. SmartGit offers a graphical access to Git repositories and can access Subversion repositories. In HarperDB, SmartGit is tracking the root folder, all subfolders and files; this includes âĂŸhdbâĂŹ which is HarperDBâĂŹs main installation folder. The schema folder contains the main database; the database layout and the most volatile data. The trash, staging, log, doc and backup folders are empty. The config file contains the default settings provided for HarperDB. Finally, the keys folder contains two public certificate files.

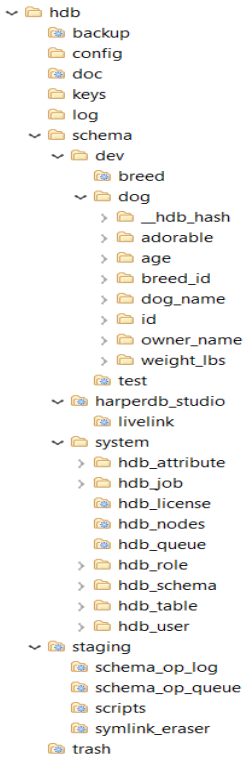

Figure 1: HarperDB tree layout

4.2 Settings, logging and testing

As depicted in Figure 2, HarperDB is using the open source WINSTON logger[2]. The default level is set to âĂŸerrorâĂŹ which results in only the most severe events being logged. There is also the option to use PINO logger[3] which is a lower overhead logging system. It can be assumed that this is what IoT devices would use via HarperDB.

[1]https://www.syntevo.com/smartgit/
[2]https://www.npmjs.com/package/winston
[3]https://www.npmjs.com/package/pino

Figure 2: HarperDB default settings

4.3 Breadcrumb Collection Process

Starting with the default settings shown in Figure 2 and the developer guides on HarperDB website [5], snapshots will be taken detailing the trail left behind by the common database actions. For example, insert, update, delete, drop, select, create and bulk load. The settings will then be modified; changing the log level and also the logging software used. This will allow us to compare and contrast the different settings/loggers from the produced dataset in the analysis phase. At each step in this process, the database and system will be restored back to the previous image using the implemented version-control system. This will ensure that each action that is taken and any changes made to configuration files do not impact the findings of this investigation.

4.4 Stored Database Data

HarperDB uses an exploded data model using operating systems files and folders, it also uses hard-links to remove duplicated data [5]. For example, the attributes of the data are stored in the folders such as “name", “age", “company" under documents named “1.*hdb*”, “2.*hdb*”, “3.*hdb*”, and so forth. The files inside the “*_hdb_hash_*” folder are commonly called “1.*hdb*”, “2.*hdb*”, “3.*hdb*” (the same as the attributes files). However, these are the primary key values of the data and they are being hard-linked directly to the attribute documents. This allows HarperDB to respond to API calls using two types of indexing; primary key and attribute without duplicating the data. A side effect of this structural model is that the data is readable within the operating system or command line, without the use of additional software.

4.5 Log Collection

As shown in Figure 2, the log file location is set as “*LOG_PATH*”, this can be changed to any directory or file name. The default permissions for the log files is 644; this means anyone can read the file but only the owner can write to it. There is only this singular log file in use for all DBMS actions. However, many of HarperDBâĂŹs competitors use a number of different logs, such as, an error log, a transaction log, a server log, [8]. All log entries contain the level

```
rob@Rob:~/hdb$ sudo ls -l schema/dev3/dog/
total 0
drwxr-xr-x 1 root root 512 Dec 20 21:25 __hdb_hash
drwxr-xr-x 1 root root 512 Dec 20 21:27 adorable
drwxr-xr-x 1 root root 512 Dec 20 21:27 age
drwxr-xr-x 1 root root 512 Dec 20 21:27 breed_id
drwxr-xr-x 1 root root 512 Dec 20 21:27 dog_name
drwxr-xr-x 1 root root 512 Dec 20 21:27 id
drwxr-xr-x 1 root root 512 Dec 20 21:27 owner_name
drwxr-xr-x 1 root root 512 Dec 20 21:27 weight_lbs
rob@Rob:~/hdb$ sudo ls -l schema/dev3/dog/id/1
total 0
-rw-r--r-- 1 root root 102 Dec 20 21:25 1545341154362.hdb
-rw-r--r-- 1 root root 102 Dec 20 21:28 1545341299682.hdb
```

Figure 3: HarperDB file access permissions

at which the entry is produced, the log entry details/message and the timestamp of the action. Figure 4 shows an example of the log with entries from the WINSTON logger with the log level set to âĂIJinfoâĂİ.

5 HARPERDB: SECURITY REVIEW AND POTENTIAL VULNERABILITIES

In the following, we start by reviewing the main security features supported by HarperDB before analysing them. We then present a set of possible enhancements aiming at improving the security of HarperDB.

5.1 Security Features Review

5.1.1 Data in Motion. As depicted by Figure 2, HarperDB supports HTTPS and CORS. HTTPS is supported using a certificate and a private key PEM file. CORS can be enabled or disabled and has support for a whitelist. CORS relates to cross-origin requests, by enabled CORS and using the whitelist, it is possible to only allow specific hosts access to the database via the API. HTTP is enabled by default but support for HTTPS can be enabled so that requests and responses are secured via SSL.

5.1.2 Authorization. The API authorization is made via "Basic Auth". There are no settings available to change this authorization type. The folders contained inside HarperDB mostly have the access permission octal 775 and the files are 644 (Figure 3). This means the folders can be read, written and executed by the owner and groups but only read and executed by other users. The files can be read and written by the owner only, but groups and other users can read the files.

5.1.3 Backup and Recovery . There is no automatic backup or recovery assistance utility built into HarperDB. All users data is stored in the "*\hdb\schema\system\hdb_user*" directory, as shown in Figure 5. The passwords and usernames are stored as folder names. The usernames are stored as plaintext and the passwords are encrypted and then stored as base64. Roles are also stored as folders with the users of a particular role stored within that folder as a file. There are no security features aimed at IoT usage of this DBMS. HarperDB does offer an application called HarperDB Studio. The studio offers a dashboard with multiple pages relating to security, health, logs and the current schema. This tool shows tracking for active nodes, disk utilisation, errors, failed login attempts, log activity (with the ability to download them to a comma-separated value sheet) and a data model view of the database's schema.


```
{"level":"info","message":"Master 173 is running","timestamp":"2018-12-22T16:46:13.585Z"}
{"level":"info","message":"In express/mnt/x/harperdb/bin","timestamp":"2018-12-22T16:46:14.358Z"}
{"level":"info","message":"Running with NODE_ENV set as: production","timestamp":"2018-12-22T16:46:14.358Z"}
{"level":"info","message":"HarperDB 1.2.004 HTTP Server running on 9925","timestamp":"2018-12-22T16:46:14.416Z"}
{"level":"info","message":"HarperDB is already running.","timestamp":"2018-12-22T16:46:14.936Z"}
```


Figure 4: HarperDB log file example

```
- active
- - true
- - - 123456789.hdb
- - - HDB_ADMIN.hdb
- - - admin.hdb
- - - qwertyuio.hdb
- - - rob.hdb
- password
- - 5i62ZSWIX4cda3a7bfdc51020634a71d5332af8dc
- - 9fnBdsHT092ca338ed6580ab8e9ad7c87c1e3a015
- - - qwertyuio.hdb
- - FmPRu5mtFc98e27b6ae54ca73921c132960420813
- - - HDB_ADMIN.hdb
- - Gu7bAAaWDebec5a933967ed3234b1d721c9015912
- - JvmDMtA4v673bc952b638f9cc3b38d8f8ab4a3c58
- - OfObpLIwd0ac503b6470e946773147a192219b832
- - - admin.hdb
- - hZ6qaNk0ub947c2915f49f6fdf433f199e95dc688
- - lEMOSUby8edc35966c265a70cbda92cdb0a8567c2
- - - 123456789.hdb
- - pGV8fbjny162cb758b6cbff6cc9aea245c9a15f92
- - xAKXzsW4ec2d4e21602836a531700ac93d27706e6
- - - rob.hdb
- role
- - c0afc808-eb3b-421f-8a9d-f6ad38fe4241
- - - rob.hdb
- - ced0f827-98a8-4eb5-9ee4-6f1ca7b6b6c6
- - - admin.hdb
- - e4055a59-1fc1-4adf-ad86-f94d03231293
- - - 123456789.hdb
- - - HDB_ADMIN.hdb
- - - qwertyuio.hdb
- - fe5b93aa-a3d7-459b-bfd4-9810efb82412
- username
- - 123456789
- - - 1545331622040.hdb
- - HDB_ADMIN
- - - 1545324695821.hdb
- - admin
- - - 1545344114290.hdb
- - qwertyuio
- - - 1545331696162.hdb
- - rob
- - - 1545342557223.hdb
- - - 1545342793918.hdb
- - - 1545344034194.hdb
- - us3r
- - - 1545331135256.hdb
- - - 1545331152234.hdb
- - - 1545331289277.hdb
- - - 1545335693646.hdb
```

Figure 5: HarperDB User Data

5.1.4 Archiving. There is no active archiving support despite the Winston logger having such a feature. By duplicating data entries in the log file, the DBMS was tested with the log file size being around 138 MB. HarperDB continued to use it and added more log entries to the end of the file during testing. There was a clear slow down (visual) in writing to the log file, system utilisation increased and performance decreased. Further research into Pino logger has concluded that log rotation is supported but is not enabled in this DBMS.

5.1.5 Data trails. Most data entries made to HarperDB are shown in plaintext in the filesystem, as shown in Figure 1 and Figure 5. If an attacker was to gain access to a server or an IoT device running HarperDB, even with basic read-only access they will be able to visually see:

- The data model layout.
- The roles that exist.
- The users that exist.
- Which users belong to which roles.
- All data entries made in the schema.

This method to catalogue data is quite unique and flexible in terms of the support that HarperDB offers in relation to both a NoSQL/SQL database. However, more security measures need to be done to adequately protect this plaintext data from being harvested and used by a malicious user to attack the DBMS.

5.2 Analysis of HarperDB's Security Features

As HarperDB only supports "Basic Auth", using this type of authentication method over HTTP means that the password can be captured and converted to plain text (man-in-the-middle attack). The attack window is also very large as every request requires authentication. Caching can also be a problem, in relation to the server or the browser sending the request.4Enabling HTTPS may reduce these attack vectors.

As explained in subsection 4.5, the "INFO" level for logging is the best available, anything beyond this produces no more log entries and anything below this level does not track the DBMS transactions taking place. Several problems exist with the best available logging level setting:

- It does not track which user is making the transactions
- Bulk uploads show as "[object Object]" in the log
- Some data trails/errors are recorded as "undefined"
- There are no log entries when HarperDB stops running

These problems are fairly critical in the creation of a meaningful audit trail, without them the logging functionality can only be considered trivial at best. Enabling Pino logger in the settings file changes the logging system that is used. The Pino logger is not enabled by default and is much lighter weight than Winston. The logs created from this tool are even less rigorous than that of Winston. There are also errors in what is reported in the logging messages. The logged data at each level does not reflect that of what Winston produces. There appears to be no thought put into this logging system and the documentation is also lacking. Many of the changes needed in the settings to test these loggers thoroughly required research into the Winston and Pino separately. This was because HarperDB offered no documentation, guide or information about these logging systems and how they are used.

Anomalies found: In the settings file, if the user changes a certain value to an incorrect parameter; for example, the user misspells the "*LOG_LEVEL*" they want to use. HarperDB will appear to be running normally, but it actually has failed to load without any output of the error. When a new user is added to the DBMS until HarperDB is restarted that user account cannot use the API. Although the user may appear in the list of created users, the API response states that the user does not exist. HarperDB supports two different tools for logging, however, both of these are incorrectly set up. There are clear errors and missing data in the audit trails produced. The default file and folder permissions set by the installation scripts of HarperDB are too open. Comparing them to the permissions set by MySQL, we see a variety of different permissions, owners, and special flags [19]. HarperDB uses the same permissions for all files and folders, irrespective of what is being stored.

5.3 Possible Security Improvements

There are several tools/functions that are missing from HarperDB, such as:

- Backup and recovery assistance.
- Security features for IoT devices.
- More authorisation types.

Most important of these would be the security features for IoT. HarperDB is expected to be used on a server and IoT devices simultaneously to allow syncing and sharing of data. It would be expected with these two vastly different use cases that the DBMS would have a toggle in the settings between them. This toggle would enable or disable extra security features that are necessary for an IoT device implemented in the wild versus a server located in a secure building.

As explained in subsection 5.1.5, the readability of the data within HarperDB has been mentioned. Taking this into consideration with the known access permissions noted in section 5.1, the conclusion would be that these access permissions are incorrectly set. The default access permission octals 775 and 644 are used for folders and files respectively, as shown in Figure 3. This means that all groups and other users have the capability to read the schema layout, usernames, data and so forth. It is possible to use access permissions that prevent this accessibility, but these are not set by default.

Moving on to HarperDB Studio; an extra utility that offers a dashboard to administrators. There are several sections here that could prove helpful from the perspective of a security audit. On the Security page, it shows active users and their roles, it also details the amount of failed login attempts that a specific user account has made. This does not track failed login attempts made via API calls.

There is also a "Health" page that states "Coming Soon!", it tracks active nodes, disk utilisation and recent errors. The recent errors list says "error 1", "error 2", etc, there are no further details. Finally, there is a "Logs" page. The information on this page reads "No data available in table" despite the log file having transactional entries. This page (if working) does not really improve the layout or readability of the log file either.

As shown in Figure 4, the sections of data written to the log file include; the log message, the level of the logging and a timestamp. An additional column exists within the Logs page called "Name". It is not known what this refers to, however, many more columns should exist in relation to the transactional data details that are stored in the log.

Table 1: DBMS landscape comparison

	HarperDB	*MongoDB*	*MySQL*	*PostgreSQL*	*Microsoft SQL*
Authentication Type	1	4	4	11	3
Logging Systems	2	1	1	1	1
Log Rotation		✓	✓	✓	✓
Log Documentation		✓	✓	✓	✓
Default Permissions	Weak	Strong	Mediium	Strong	Strong
Backup Functionality		✓	✓	✓	✓
Recovery Function		✓	✓	✓	✓
Encryption		✓	✓	✓	✓
Dashboards	1	Many	Many	Many	Many
Archiving Support		✓	✓	✓	✓
Compression		✓	✓	✓	✓
User Tracking		✓	✓	✓	✓

5.4 DBMS landscape comparison

In Table 1 we provide a comparative analysis of different DBMSs, including *HarperDB*, *MongoDB*, *MySQL*, *PostgreSQL*, and *Microsoft SQL*.

In MongoDB, there are four different authentication methods available, i.e., SCRAM, x.509, LDAP and Kerberos [6], while HarperDB supports only one. The default file and folder permissions set by the installation scripts of HarperDB are too open. Comparing them to the permissions set by MySQL, we see a variety of different permissions, owners, and special flags [7]. While, HarperDB uses the same permissions for all files and folders, irrespective of what is being stored. Considering the data model structure of this DBMS in relation to the folder and file names containing detailed information about the data in the database, it is expected that the permissions would be more stringent than those set by MySQL.

6 BEST PRACTICES FOR END-USER AND DEVELOPERS

Based on what we discussed so far, it is evident that the setting of HarperDB has some flaws that could be misused to perform several attacks. In this section, based on our review on HarperDB 1.2.004, we provide best practice recommendations for end-users and developers of HarperDB.

End-user advice: Based on our discussion of the logging issues in HarperDB, we advise the users to: 1) use the Winston logger and set the logging level to âĂIJINFOâĂİ; 2) Change the permissions of files and folders to the octals 600 and 700, respectively; 3) Try to have only one user, as tracking more than one is currently not possible; 4) Turn HTTPS on and create a CORS whitelist that supports your host only; 5) Create a script that will change the log file name in âĂIJsettings.jsâĂİ periodically. This script could also archive and backup these log files; 6) Change the ports and admin username from their default settings in case bad actors are intentionally snooping and using HarperDBâĂŹs known default settings as an attack vector; 7) Store the PEM keys use for SSL outside of the HarperDB (HDB) folder.

Development advice: To address the issues that we pointed out in this paper, we recommend the following guidelines to be taken into consideration. 1) The file and folder permissions should be reconsidered. Focusing on the data that HarperDB is storing as plaintext as part of its data model structure, envisage an internal threat. 2) It would be better to add user tracking to the logs. Irrespective of whether a user is using a select statement, an insert or modification statement, their actions should be logged thoroughly, with the user account in questions query being logged and the results they received. 3) It is required to consider auditing, especially in the case of a forensic analyst performing an investigation on the DBMS. 4) A process for log cycling and archiving should be in place. Tests may need to be performed to find what file size is too big. Perhaps certain logged data is less important than other data, the creation of multiple log files for these varying entries may prove useful. 5) It would be better to have automatic IoT log syncing to the server and resetting. If an IoT device was tampered with and the data in the database had already been moved to the server and cleared, it would be unfortunate to not have cleared the log of the same data. 6) The bulk upload message response should be fixed, so that the object array is output in a manner by which the data can be read, analysed and re-used in a recovery scenario. 7) It would be better to have more authentication types, while thinking about IoT tampering and the idea of rogue devices being used by external threat actors. How would this impact the main server running HarperDB, how could this impact the data and how can it be prevented. 8) There should be systems for backup and recovery, envisage broken IoT devices being replaced and the work involved in adding new ones. Cloning devices would prove useful, how could this work in a secure way.

7 CONCLUSION

HarperDB offers IoT devices more functionalities and is easy to use, but, it should only be considered for non-sensitive applications in its current version. Indeed, if a company is attacked with HarperDB's default settings, they would not be able to; track the breached account, track tampered or removed data or understand what data has been stolen. There are currently a variety of attack vectors possible to stay hidden from the logging system within HarperDB. Therefore, additional software should be recommended to users are going to use this DBMS in a live environment and

guides should exist that explain all of the positive and negative aspects of using HarperDB from a security point of view. IoT devices are well known for being used in insecure environments, HarperDB adds nothing to enhance these inherently weak devices. The storage of sensitive information in HarperDB is also of concern. The data model structure that uses folders and files is unique, fast and small. However, there exists a flaw that if data are not encrypted prior to storage then they are visible in plaintext to read via anyone who has access to the Linux machine. It is expected that documentation about this should be provided and the concerns regarding securing data properly in HarperDB are explained thoroughly, which it is not done currently.

REFERENCES

[1] 2018. Bridgwater, A.. No Fleas On HarperDB, The IoT Database Ready To 'Go Fetch' At The Edge. https://www.forbes.com/sites/adrianbridgwater/2018/07/10/no-fleas-on-harperdb-iot-database-ready-to-go-fetch-at-the-edge.

[2] 2018. Cassandra Database. http://cassandra.apache.org..

[3] 2018. GHarperDB. Our Story. https://www.harperdb.io/our-story.

[4] 2018. Goldberg, S. Insights and Updates. https://www.harperdb.io/blog/harperdb-for-iot-launches-today.

[5] 2018. HarperDB. Simplifying Big Data Architecture - HTAP Database. https://www.harperdb.io.

[6] 2018. MongoDB Security Concept. http://docs.mongodb.org/master/core/security/.

[7] 2018. MySQL Developer Documents. (n.d.). Chapter 11 Enabling Authentication. https://goo.gl/YqXrBz.

[8] Oluwasola Mary Adedayo and Martin S Olivier. 2015. Ideal log setting for database forensics reconstruction. *Digital Investigation* 12 (2015), 27–40.

[9] Arafat Al-Dhaqm, Shukor Razak, Siti Hajar Othman, Kim-Kwang Raymond Choo, William Bradley Glisson, Abdulalem Ali, and Mohammad Abrar. 2017. CDBFIP: Common database forensic investigation processes for internet of things. *IEEE Access* 5 (2017), 24401–24416.

[10] Arafat Mohammed Rashad Al-Dhaqm, Siti Hajar Othman, Shukor Abd Razak, and Asri Ngadi. 2014. Towards adapting metamodelling technique for database forensics investigation domain. In *2014 International Symposium on Biometrics and Security Technologies (ISBAST)*. IEEE, 322–327.

[11] Sana Belguith, Shujie Cui, Muhammad Rizwan Asghar, and Giovanni Russello. 2018. Secure publish and subscribe systems with efficient revocation. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. ACM, 388–394.

[12] Sana Belguith, Nesrine Kaaniche, and Mohammad Hammoudeh. 2019. Analysis of attribute-based cryptographic techniques and their application to protect cloud services. *Transactions on Emerging Telecommunications Technologies* (2019), e3667.

[13] Sana Belguith, Nesrine Kaaniche, and Giovanni Russello. 2018. PU-ABE: lightweight attribute-based encryption supporting access policy update for cloud assisted IoT. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE, 924–927.

[14] Sana Belguith, Nesrine Kaaniche, Giovanni Russello, et al. 2018. Lightweight attribute-based encryption supporting access policy update for cloud assisted IoT. In *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications-Volume 1: SECRYPT*. SciTePress, 135–146.

[15] Shujie Cui, Sana Belguith, Pramodya De Alwis, Muhammad Rizwan Asghar, and Giovanni Russello. 2018. Malicious entities are in vain: Preserving privacy in publish and subscribe systems. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 1624–1627.

[16] Shujie Cui, Sana Belguith, Ming Zhang, Muhammad Rizwan Asghar, and Giovanni Russello. 2018. Preserving Access Pattern Privacy in SGX-Assisted Encrypted Search. In *2018 27th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 1–9.

[17] Shrankhla Gupta, Nikhil Kumar Singh, and Deepak Singh Tomar. 2018. Analysis of NoSQL Database Vulnerabilities. (2018).

[18] Werner K Hauger and Martin S Olivier. 2017. Forensic attribution in NoSQL databases. In *2017 Information Security for South Africa (ISSA)*. IEEE, 74–82.

[19] Neal Leavitt. 2010. Will NoSQL databases live up to their promise? *Computer* 43, 2 (2010), 12–14.

[20] Hossain Shahriar and Hisham M Haddad. 2017. Security Vulnerabilities of NoSQL and SQL Databases for MOOC Applications. *International Journal of Digital Society (IJDS)* (2017).